

# Model Checking Degrees of Belief in a System of Agents

Franco Raimondi  
Middlesex University  
F.Raimondi@mdx.ac.uk

Giuseppe Primiero  
Middlesex University  
G.Primiero@mdx.ac.uk

Neha Rungta  
NASA Ames Research Center  
neha.s.rungta@nasa.gov

## ABSTRACT

Reasoning about degrees of belief has been investigated in the past by a number of authors and has a number of practical applications in real life. In this paper we present a unified framework to model and verify degrees of belief in a system of agents. In particular, we describe an extension of the temporal-epistemic logic CTLK and we introduce a semantics based on *interpreted systems* for this extension. In this way, degrees of beliefs do not need to be provided externally, but can be derived automatically from the possible executions of the system, thereby providing a *computationally grounded* formalism. We leverage the semantics to (a) construct a model checking algorithm, (b) investigate its complexity, (c) provide a Java implementation of the model checking algorithm, and (d) evaluate our approach using the standard benchmark of the dining cryptographers. Finally, we provide a detailed case study: using our framework and our implementation, we assess and verify the *situational awareness* of the pilot of Air France 447 flying in off-nominal conditions.

## 1. INTRODUCTION

Suppose you draw a seven of diamonds from a deck of cards, and your friend Alice draws another card that she keeps secret. You obviously *know* that you have a seven of diamonds, and you obviously do not know Alice's card. However, you can *believe* that Alice has an ace of spades (nothing rules out this possibility). You can also believe that Alice has a card whose suite is hearts. It also seems natural to think that the latter belief has a "greater weight" than the former or, equivalently, that you have a "greater degree of belief" in the latter.

A standard approach to belief quantification involves the use of *probabilities* and the example of cards described above is interpreted in terms of probabilities by almost all readers. However, beliefs can be quantified using a number of other approaches (see [13] for a detailed overview). One way to characterise this literature is by referring to *objective* and *subjective* assignments to degrees of belief. Subjective

assignments differentiate between actual probabilities and agents' beliefs, while objective assignments refer to actual features in the real world (for instance when modelling a biased coin). In this paper we employ the term *degrees of belief* and we avoid references to probabilities, thereby taking what could seem a *subjective* approach. Nonetheless, there is a connection between our approach to modelling degrees of belief and probability distributions; this link will become clear after the introduction of our technical machinery and we will return to this connection in Section 6. For the time being, however, we ask the reader to avoid interpreting the weight of doxastic modalities in terms of probabilities, as our aim here is to introduce a unified framework to model and verify degrees of belief in a system of agents. More in detail, our contributions can be summarised as follows:

- We provide a *computationally grounded* formalism to reason about degrees of belief by introducing an extension of the logic CTLK whose semantics is based on interpreted systems [9]. We name this extension COGWED: a COmputationally Grounded, wEighter Doxastic Logic.
- We introduce a model checking algorithm for COGWED by extending the standard algorithm for CTLK, together with its complexity analysis.
- We implement and we release as open source a model checker for COGWED and we use the benchmark of the dining cryptographers to prove the feasibility of our approach.
- We employ our model checker to verify the key properties of a safety-critical scenario.

The rest of the paper is organised as follows: in Section 2 we review the formalism of interpreted systems; in Section 3 we present COGWED and its model checking algorithm; in Section 4 we introduce a model checker, its implementation and its performance evaluation on the protocol of the dining cryptographers. Finally, in Section 5 we introduce a motivational example where we show how our approach can be used to characterise the *situational awareness* of a pilot flying in off-nominal conditions. In particular, we consider the model of the Air France 447 accident provided in [1] and we evaluate the situational awareness of the pilot when a stall occurs. We show that there exist cases in this model in which the plane is actually stalling, but the pilot has a very low degree of belief about the stall, a situation that can be formally analysed with our tool.

**Appears in:** Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), Lomuscio, Scerri, Bazzan, Huhns (eds.), May, 5–9, 2014, Paris, France.

Copyright © 2014, International Foundation for Autonomous Agents and Multiagent Systems ([www.ifaamas.org](http://www.ifaamas.org)). All rights reserved.

## 2. PRELIMINARIES

### 2.1 Interpreted Systems

We employ here the formalism of *Interpreted Systems* from [9] to describe a system of agents.

In particular,  $IS = (G, R_t, V)$  where

- $G = \bigtimes_{i=1}^n L_i$  is a finite set of *global states*, obtained as the cartesian product of  $n$  sets of *local states* (one set for each agent);
- $R_t \subseteq G \times G$  is a temporal relation (it is assumed that each state has at least a successor);
- $V : AP \rightarrow 2^G$  is an evaluation function for atomic propositions;

The formalism of interpreted systems presented in [9] and employed in other model checkers such as [18, 12] also includes the notions of agents' actions and agents' protocols: to keep our presentation simple, we do not consider these here, as they play no role in the semantics for the logic presented below.

We define a set of  $n$  *equivalence relations* (one for each agent): let  $g = (l_1, \dots, l_n)$  and  $g' = (l'_1, \dots, l'_n)$  be two global states from  $G$ ; we define  $gR_i g'$  iff  $l_i = l'_i$ , i.e., two global states  $g, g'$  are equivalent for agent  $i$  iff the local state of agent  $i$  is the same in  $g$  and in  $g'$  (notice that these are the standard epistemic relations used in [9] to interpret epistemic modalities). We define  $\{g\}_{R_i}$  to be the equivalence class of the global state  $g$  w.r.t.  $R_i$ .

Given an interpreted system  $IS$  and a global state  $g$ , logic formulas involving CTL and epistemic operators can be interpreted as follows (we refer to [9] and references therein for more details about CTL syntax and semantics):

$IS, g \models p$	iff	$g \in V(p)$ ;
$IS, g \models \neg\varphi$	iff	$IS, g \not\models \varphi$ ;
$IS, g \models \varphi \wedge \psi$	iff	$IS, g \models \varphi$ and $IS, g \models \psi$ ;
$IS, g \models EX\varphi$	iff	there exists $g' \in G$ s.t. $gR_t g'$ and $IS, g' \models \varphi$ ;
$IS, g \models EX\varphi$	iff	there exists $g' \in G$ s.t. $gR_t g'$ and $IS, g' \models \varphi$ ;
$IS, g \models EG\varphi$	iff	there exists a path $\pi = (g, g_1, \dots)$ such that, for all $i$ , $g_i R_t g_{i+1}$ and $IS, g_i \models \varphi$ ;
$IS, g \models E[\varphi U\psi]$	iff	there exists a path $\pi = (g, g_1, \dots)$ and an index $j$ such that $IS, g_j \models \psi$ and $IS, g_i \models \varphi$ for all $i \leq j$ ;
$IS, g \models K^i\varphi$	iff	$gR_i g'$ implies $IS, g' \models \varphi$ .

With slight abuse of notation we denote with  $V(\varphi)$  the set of states of an interpreted system  $IS$  in which  $\varphi$  holds. This logic is usually named CTLK and can include group epistemic modalities to reason about distributed and common knowledge. In the next section we will extend this logic with doxastic operators.

### 2.2 Model checking Interpreted Systems

Given a logic formula  $\varphi$  and an appropriate model  $M$  for  $\varphi$ , in general terms model checking is the problem of establishing whether or not  $M \models \varphi$ , usually in an automated way. In the context of Interpreted Systems, model checking is the problem of verifying that a given CTLK formula  $\varphi$  holds in all the global states of an Interpreted System  $IS$ .

The complexity of model checking CTLK formulae in a given Interpreted System is polynomial in the size of the model and formula [6, 9]. The standard algorithm operates recursively on the structure of the formula by “labelling” the global states of the Interpreted System with the sub-formulae that are true there. We refer to [6, 9] for additional details.

We remark here that, in many cases, the model is generated from a succinct description by means of model variables; in this case, adding a simple Boolean variable causes the model to double in size. This is known as the *state explosion problem* and the complexity of model checking CTLK formulae against succinct representations requires deterministic algorithms that have an exponential complexity in the size of the representation [19]. Symbolic algorithms using Ordered Binary Decision Diagrams and reduction to SAT problems have been successfully employed in various tools [18, 12] for multi-agent system verification to tackle this complexity. We return to this issue in Section 3.3.

## 3. MODEL CHECKING COGWED

In this section we introduce the syntax of COGWED, its semantics with some key equivalences and a model checking algorithm for it. We also present some complexity considerations.

### 3.1 COGWED Syntax and Semantics

Let  $\sim$  be one of the following comparison operators:  $\{<, \leq, =, \geq, >\}$ . The syntax of COGWED is as follows:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \psi \mid B_{\sim x}\varphi \mid EX\varphi \mid EG\varphi \mid E[\varphi U\psi] \mid K_i\varphi$$

Where:

- $p$  is an atomic proposition from a set  $AP$ ;
- $i$  is an index for agents, ranging from 1 to  $n$ ;
- $x$  is a real number,  $0 \leq x \leq 1$ ;
- $EX\varphi, EG\varphi, E[\varphi U\psi]$  are standard CTL temporal operators.
- $K_i$  is the standard epistemic operator.

Essentially, COGWED extends CTLK with the additional operators  $B^i$  (one for each agent) and with comparison operators. The formula  $B_{\sim x}\varphi$  is read as “Agent  $i$  believes  $\varphi$  with a degree of belief  $\sim x$ . For instance,  $B_{\leq 0.2}^2(p \vee q)$  is read as “Agent 2 believes  $(p \vee q)$  with a degree of belief less or equal than 0.2, and  $B_{= 0.5}^2(B_{\leq 0.1}^1(p))$  means that “Agent 2 believes with degree exactly equal to 0.5 that Agent 1 believes with degree at most 0.1 that  $p$ ” (where  $p$  could mean “agent 2 has an ace of space”). As we will see below,  $B_{\sim 1}^i\varphi$  is equivalent to  $K_i\varphi$ . COGWED formulae are evaluated in Interpreted Systems by extending the definitions provided in the previous section with the following:

$$IS, g \models B_{\sim x}\varphi \quad \text{iff} \quad \frac{|V(\varphi) \cap \{g\}_{R_i}|}{|\{g\}_{R_i}|} \sim x$$

The intuition behind this definition is the following: the degree of belief that an agent associates to a formula  $\varphi$  in a global state  $g$  is the ratio between the number of states of  $\{g\}_i$  (the equivalence class of  $g$ ) in which  $\varphi$  is true and the total number of states in  $\{g\}_i$ . For instance, considering again the scenario in which you draw a seven of diamonds from a deck of card, and Alice draws another card that she

```

1 // We associate a set of equivalence
2 // classes to each agent:
3 Map <Integer , Set <Set<Gstate>>> rk;
4
5 // This method computes the set of
6 // states in which  $B_{\sim x}^i f$  is true
7 public Set<Gstate> satB(int i, Formula f,
8                           String op , float x) {
9     Set<Gstate> previous = SAT(f);
10    Set<Gstate> result = new Set();
11    for (Set<Gstate> eqClass: rk.get(i)) {
12        if ( [eqClass∩previous] ~ x ) {
13            result.add(eqClass);
14        }
15    }
16    return result;
17 }
```

Figure 1: Java-style algorithm sketch

keeps secret. If the deck has 52 cards overall, your belief about the fact that Alice has an ace of diamonds has a degree of 1/51, and your belief that Alice has a card whose suite is hearts has a degree of 13/51. As a result, the following formula is true:

$$B_{\leq 0.05}^{\text{Reviewer}}(\text{Alice\_ace\_spade}) \wedge B_{\geq 0.2}^{\text{Reviewer}}(\text{Alice\_hearts})$$

This definition of degrees of beliefs is *computationally grounded* in the sense of Wooldridge [23]: modalities are interpreted directly on the set of possible computations of a multi-agent system (equivalently: modalities are interpreted on a Kripke model that corresponds to the possible computations of a multi-agent systems), and there is no need to provide weights as part of the model. We refer to Section 6 for a comparison with other existing approaches to evaluate degrees of belief.

The following formulas are valid in all COGWED models as a result of simple arithmetic considerations:

1.  $B_{\leq x}^i \varphi \rightarrow B_{\leq y}^i \varphi$  for all  $y \geq x$ ;
2.  $B_{\geq x}^i \varphi \rightarrow B_{\geq y}^i \varphi$  for all  $y \leq x$ ;
3.  $B_{\geq x}^i \varphi \leftrightarrow B_{\leq (1-x)}^i \neg \varphi$

Finally, it is easy to see that  $B_{=1}^i \varphi$  is equivalent to  $K_i \varphi$ , i.e., a degree of belief equal to 1 corresponds to the standard epistemic operator. Dually, as a result of the third formula above, it is also true that  $B_{\leq 0}^i \varphi \leftrightarrow K^i(\neg \varphi)$ .

## 3.2 The algorithm

In this section we describe a model checking algorithm for the operator  $B_{\sim x}^i f$ . We do this by describing a method **satB** that can be included in the standard model checking algorithm for CTLK. A Java-like description of the algorithm is provided in Figure 1.

The method employs the set of equivalence classes for each agent; this set can be computed by partitioning the set of global states (remember that each global state is a tuple of local states). The result of this operation is the map **rk** (line 3), which associates an agent ID (in the form of an Integer variable) to a set of sets of global states (i.e., the set of equivalence classes).

The method **satB** returns the set of global states satisfying the formula  $B_{\sim x}^i f$ . It starts by (recursively) calling a

method **SAT(f)** that computes the set of states in which the formula  $f$  is true (line 9). Then, it iterates over the equivalence classes of agent  $i$  (line 11). In line 12 the method computes the ratio of the set in which the formula is true in a given equivalence class over the size of the actual equivalence class. If this ratio satisfies the appropriate relation  $\sim$ , then the method adds the *whole* equivalence class to the set of states in which the formula is true (line 13). The intersection of sets of states can be performed with standard library functions provided by Java; we refer to the source code available online for additional details about the actual implementation. The final result is returned at line 16.

As mentioned above, notice that the algorithm does not operate on individual states. Instead, once the equivalence classes are built, the algorithm works with *sets* of states. We investigate the complexity of this algorithm in the next section.

## 3.3 Complexity considerations

Model checking CTLK formulae in an interpreted system takes time polynomial in the size of the formula and in the size of the model [9]. All the operations in the algorithm described in Figure 1 require at most polynomial time: computing the set of equivalence classes, iterating over them, and computing intersection of states. Therefore, the method described above remains in the same polynomial complexity class of the standard CTLK model checking algorithm.

As mentioned in Section 2.2, in practical applications the actual state space is likely to explode as a result of the number of variables employed to model a given scenario. A number of techniques are available to manage large state spaces. In particular, Ordered Binary Decision Diagrams (OBDDs) are employed in model checkers for multi-agent systems such as MCMAS [18] and MCK [12]. The algorithm of Figure 1 operates on set of states from line 7 to line 17 and only performs intersections of sets: these operations can be performed on the OBDDs for the sets of states, and therefore this part of the algorithm can be executed symbolically. The computation of equivalence classes needed at line 3, however, may require in the worst case the explicit enumeration of all reachable states, if all global states are epistemically different for a given agent. This is rarely the case and, in fact, the number of equivalence classes is normally orders of magnitude smaller than the number of global states. This is indeed the case in the examples that we present below in Section 4.2 and in Section 5.

## 4. MC-COGWED: A TOOL TO VERIFY COGWED PROPERTIES

In this section we describe a model checker for the verification of COGWED properties, called Mc-COGWED. This is a prototype implementation that is used to evaluate the algorithm presented above on the standard example of the dining cryptographers. All the source code, the benchmarks of the dining cryptographers and the card examples, and a pre-compiled version are available from this link: <https://sites.google.com/site/mccogwed/>

### 4.1 Implementation overview

Mc-COGWED is implemented entirely in Java. The input language of the model checker is a simple description of the states and transitions in a model. An example of this

```

1 // Just two agents
2 N=2;
3
4 // The list of global states
5 S1 = (c1,c2);
6 S2 = (c1,c3);
7 S3 = (c2,c1);
8 S4 = (c2,c3);
9 S5 = (c3,c1);
10 S6 = (c3,c2);
11
12 // If needed, a temporal relation
13 // can be specified using the following syntax:
14 // RT = { (S1,S2),(S1,S3),... };
15
16 // The labelling function:
17 agent1_has_card1 = { S1, S2 };
18 agent2_has_card1 = { S3, S5 };
19 // [...]

```

Figure 2: Input file for Mc-COGWED (3 cards)

language is provided in Figure 2: this example describes a scenario with 2 agents and all the possible states resulting from the agents picking a card from a deck with three cards. More in detail: in line 2 we specify the number of agents. The lines from 5 to 10 encode the set  $G$  of global states; each global state is identified by an ID (S1 to S6) and is described by a pair of local states as there are only 2 agents in this example. For instance,  $S3 = (c2, c1)$  corresponds to the global state in which the local state for the first agent is  $c2$  and the local state for the second agent is  $c1$ , i.e., the global state in which the first agent has card 2 and the first agent has card 1. We do not include a temporal relation for this simple example but, as exemplified in the comment at line 14, the temporal relation is represented by a list of pairs of states (the protocol of the dining cryptographers below contains a temporal relation). Finally, lines 17 and 18 provide an example definition of two atomic propositions. The first atomic proposition is true when agent 1 has card 1 (in global states S1 and S2), while the second proposition is true when agent 2 has card 1 (in global states S3 and S5).

This is a very simple language, but our aim here is to provide a concrete assessment of the complexity of the approach and to show that COGWED can be successfully employed in the verification of real-life scenarios (see Section 5).

Mc-COGWED parses the input file using ANTLR [20] and builds an explicit representation of the model using standard Java structures for sets and maps. The epistemic relations are automatically generated from the structure of the global states by imposing the equivalence of local states.

In addition to the input file for the model, Mc-COGWED takes a COGWED formula as an input parameter from the command line. We have implemented the model checking algorithm for the minimal set of temporal operators EX, EG and EU, for Boolean expressions and for the belief operator  $B_{\sim x}^i$ . Mc-COGWED operates recursively on the structure of the formula and generates a set of (global) states that can be either explicitly printed on screen, or the tool could simply report the number of global states where the input formula is true.

We provide a generator for the card example mentioned above in the directory `examples/` of the source code. The generator takes as an input parameter the number of cards to be generated and creates an input file similar to the one in

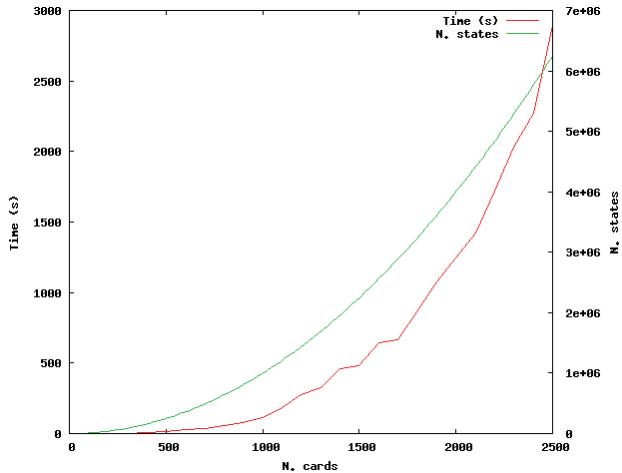


Figure 3: Performance results for the card example.

Figure 2. We report in Figure 3 the number of global states (right scale) and the execution time (left scale) for a number of cards ranging from 100 to 2500 for the verification of the formula  $\text{agent1\_has\_card1} \rightarrow (K^1(B_{<\frac{1}{(n-1)}}^2 \text{agent1\_has\_card1}))$ .

This formula expresses the fact that, if agent 1 has card 1 from a deck of  $n$  cards, then agent 1 *knows* that agent 2 *believes* with a degree of belief less than  $\frac{1}{(n-1)}$  that agent 1 has card 1: this formula is true in all the states of the model and forces the exploration of all the equivalence classes. Figure 3 shows that the tool is able to verify up to 2500 cards (corresponding to a state space of approximately 7 million states) in less than 45 minutes. The figure displays the quadratic dependance of the total number of global states on the number of cards, and a polynomial of higher degree for the verification time: this is due to the two nested epistemic and doxastic modalities together with the computation of two set intersections. These and all the results below are obtained on a Macbook Pro, 2.4 GHz Intel Core i7 with 16 GB RAM and running Mac OS X 10.8.5. Mc-COGWED was compiled using Java version 1.7.0, revision 40, and the Java virtual machine was configured with a heap size of 12 GB. A more detailed performance evaluation is carried out in the next section.

## 4.2 Performance evaluation: the dining cryptographers

In this section we conduct a more detailed evaluation of performance for Mc-COGWED using the protocol of the dining cryptographers. The protocol of the dining cryptographers is a standard benchmark in the multi-agent verification community, as it employs temporal-epistemic specifications and it can be easily scaled up. The protocol, originally described in [4], is exemplified by the following scenario: three cryptographers sit at a round table at a restaurant. A waiter informs them that the bill has already been paid for. The cryptographers now wonder whether one of them paid for the bill, or whether it was paid by their company. To preserve the anonymity of the payer, they run the following protocol: each one of them flips a coin behind a menu on their right, so that this coin is only visible by the person who flipped the coin and by the next cryptographer to the

right. In this way, each cryptographer sees two coins. After the initial round of coin tosses, each cryptographer has to announce whether s/he sees two equal coins (e.g., two heads or two tails), or two different coins. However, if the cryptographer paid for the dinner, than s/he has to say the *opposite* of what s/he sees. The key property of the protocol is that, if there is an even number of cryptographers announcing that the coins are different, then the company paid for the dinner; if the number of “different” utterances is odd, however, then someone at the table paid for dinner. In this case, it is possible to verify the key epistemic property:

$$(\text{odd} \wedge \neg \text{paid}_1) \rightarrow (K^1(\text{paid}_2 \vee \text{paid}_3) \wedge \neg K^1(\text{paid}_2) \wedge \neg K^1(\text{paid}_3))$$

which encodes the fact that, if the first cryptographer did not pay for the dinner and there is an odd number of “different” utterances, then the first cryptographer knows that either cryptographer 2 or cryptographer 3 paid for the dinner (i.e., the cryptographer knows the disjunction), but cryptographer 1 does not know that cryptographer 2 paid, nor cryptographer 3. It is also possible to verify that the same formula holds for any number of cryptographers greater than 2. In COGWED we can refine this formula and introduce a *degree of belief* for the first cryptographer, in the case s/he did not pay:

$$AG \left( (\text{odd} \wedge \neg \text{paid}_1) \rightarrow \left( \bigwedge_{i=2}^n B_{\left(=\frac{1}{n-1}\right)}^1(\text{paid}_i) \right) \right)$$

This formula captures the fact that an odd number of utterances places an *equal* degree of belief on the fact that any of the remaining cryptographers could be the payer.

We have implemented a Java generator for the dining cryptographers in Mc-COGWED; this generator is available under `examples/` in the source files and takes the number of cryptographers as an input parameter. Each cryptographer is modelled with 4 local variables: value of left and right coin (possible values: Empty, Head, Tail), whether the cryptographer is the payer (Empty, Yes, No), and the parity of “different” utterances (Empty, Even, Odd). In the initial state the value of these variables is set to empty for all cryptographers. The generator then runs the protocol by producing a random initial configuration and outputs a file in COGWED format with the set of *reachable* global states, the temporal transition relation for these states, and an appropriate labelling function for the global states. This file is then passed to Mc-COGWED, together with the formula described above.

We ran experiments with a number of cryptographers ranging from 3 to 15. Experimental results are reported in Table 1. The first column reports the number of cryptographers; the second column labelled with  $|S|$  reports the size of the state space (in our encoding of the example this is simply  $(3^4)^n$ , where  $n$  is the number of cryptographers); the third column  $|G|$  is the number of *reachable* states as computed by our generator; the fourth column  $R_t$  reports the number of pairs in the transition relation; the fifth column reports the time required to generate the set of reachable states and to write this set to a file. The final column reports the time required to parse this file and verify the formula reported above by the actual Mc-COGWED model checker. The size of the generated file exceeds 300 Mb for 16 cryptographers and this causes the ANTLR parser to run out of memory before invoking the generation of epistemic relations and the verification of the formula. In all the other cases, the overall execution time obtained by adding the generation of

N	$ S $	$ G $	$ R_t $	gen. time (s)	verif. time (s)
3	$5 \cdot 10^5$	65	96	0.11	0.12
4	$4 \cdot 10^6$	161	240	0.12	0.16
5	$3 \cdot 10^8$	385	576	0.15	0.31
6	$2 \cdot 10^{10}$	897	1344	0.18	0.33
7	$2 \cdot 10^{12}$	2049	3072	0.25	0.46
8	$1.15 \cdot 10^{15}$	4609	6912	0.38	0.49
9	$1.50 \cdot 10^{17}$	10241	15360	0.51	0.83
10	$1.22 \cdot 10^{19}$	22529	33792	0.67	1.27
11	$9.85 \cdot 10^{20}$	49153	73728	1.17	4.16
12	$7.98 \cdot 10^{22}$	106497	159744	1.94	6.74
13	$6.46 \cdot 10^{24}$	229377	344064	3.35	23.48
14	$5.23 \cdot 10^{26}$	491521	737280	6.77	70.38
15	$4.24 \cdot 10^{28}$	1048577	1572864	14.39	175.16

Table 1: Dining cryptographers: results

the reachable state space and the actual verification time remains below 4 minutes even for 15 cryptographers; we are therefore confident that a more compact representation of the example using a more expressive modelling language for COGWED models could enable the verification of even larger state spaces.

These results are very encouraging, as they are comparable to what highly optimised and symbolic model checkers such as MCMAS and MCK can achieve for standard epistemic modalities (see, for instance, the results reported in Table 2 of [18]). Our results thus show that reasoning about degrees of beliefs in a system of agents is feasible even for large state spaces, even for formulae involving both temporal and doxastic modalities.

Besides being computationally tractable, in the next section we show how model checking COGWED can have practical applications in analysing safety-critical scenarios.

## 5. CASE STUDY: THE AIR FRANCE 447 ACCIDENT

In the previous sections we have employed COGWED to characterise two scenarios that are typical in security and communication protocols. However, degrees of belief can be used to reason formally about other specification patterns. In this section we show how *situational awareness* can be assessed using COGWED. Informally, situational awareness is the ability of an agent (typically human) to assess a situation and to understand how the environment will react to the agent’s actions. Situational awareness is a key factor for decision makers in safety-critical situations, such as airplane pilots, medical doctors, firemen, etc, and it has been investigated extensively in the past in a number of research areas, including psychology [10]. Here we focus on the aeronautic domain, working in collaboration with domain experts from NASA Ames.

### 5.1 A model for AF447

The Air France flight 447 from Rio de Janeiro to Paris is a thoroughly investigated accident involving the failure of a sensor (a set of Pitot tubes), resulting in incorrect speed readings and, through a sequence of events, to a high-altitude stall situation that failed to be diagnosed by the

pilot(s). The BAE report on the accident<sup>1</sup> attributes the main cause of the accident to the inexperience of the pilot, who was not able to assess the actual speed of the airplane and, more crucially, the stall situation.

We employ here a Java simulation model of the scenario taken from [1] and we modify it to generate a set of reachable states using the approach presented in [16], in collaboration with domain experts at NASA Ames. The set of reachable states obtained is then encoded in Mc-COGWED input. We remark that our model does not aim at being an accurate representation of the accident; instead, our aim is to show the capabilities of COGWED in analysing *situation awareness*. In our model, a plane and its environment are characterised by:

- an actual external temperature (low, medium, high);
- an actual speed (very low, low, medium, high, very high);
- an actual vertical speed (Climbing, null, Descending);
- an actual altitude (encoded using flight levels, such as FL200, FL380 and FL450);
- an actual attitude (going up, flat, down);
- an actual thrust level (auto, 20%, 50%, TOGA, full. “TOGA” is an auto-thrust level corresponding to the thrust required for Take-Off or a Go-Around landing)

In the actual situation the pilot is flying in the dark: it only has direct access to the thrust, which we assume is correct. All the remaining parameters are accessed through sensors that may be faulty. As a result, we characterise the local states of the pilot by means of:

- observed temperature;
- observed speed;
- observed vertical speed;
- observed altitude;
- observed attitude.

All these values are observed by means of *sensors*, some of which may fail. When a sensor is broken, the observed value of a parameter may differ from the actual value. Additionally, a plane includes:

- an auto pilot to which the pilot has direct access, i.e., the pilot can observe whether the auto pilot is engaged or not, and we assume that the auto pilot does not fail.
- a set of Pitot tubes that may be frozen when the temperature is low (but not necessarily). If the Pitot tubes are frozen, then the speed sensor is broken (but the speed sensor could be broken even when the Pitot tubes are not frozen).

---

<sup>1</sup><http://www.bea.aero/en/enquetes/flight.af.447/flight.af.447.php>

- a stall warning (in the form of audio message or stick shaking, depending on the causes of the stall). Notice that the *stall warning disengages when the speed is very low* (below 60 kt), even if the plane could be actually stalling. We assume that the stall warning signal does not fail, i.e. a warning always corresponds to stalling conditions.

We model the behaviour of the pilot based on the procedures required in the various cases. For instance, if the observed speed is very high (a potentially very dangerous situation) the pilot reduces thrusts, and if the stall warning is on, the pilot modifies attitude and thrust appropriately. The Java simulation modifies the actual values of the airplane characteristics according to pilot’s actions and standard physics laws, generating new states every time a value changes.

To generate the set of possible states for this scenario, we start from a situation in which the plane is flying at flight level 380 (corresponding to 38,000 feet), the thrust is 60%, the auto pilot is engaged, the stall warning is off, attitude is flat, temperature is medium and all sensors are working correctly. We then inject failures in the sensors (as mentioned above, these are not diagnosable by the pilot: remember that the pilot is flying in the dark and has therefore no possibility of assessing (vertical) speed, attitude, altitude and temperature) and we generate a COGWED model covering all possible combinations reachable from the initial state. The generation is achieved by running the Java code developed in [1] and by discretizing the continuous variables where required (in this case: speed, vertical speed, attitude, altitude, temperature). The number of possible discretized states is  $2 \cdot 10^8$ , of which approximately  $1.6 \cdot 10^5$  are reachable from the initial state described above.

We can now use Mc-COGWED to evaluate the fact that the pilot is aware of a stall. In particular, we want to assess the degree of belief of a stall situation. To this end, we employ the following formula:

$$EF(actualStall \wedge B_{<0.05}^{Pilot}(actualStall))$$

This formula encodes the fact that there exists a state reachable from the initial state, such that the plane is actually stalling, but in that specific state the pilot believes that the stall is actually occurring with a degree of less than 5%: this formula is true in 25 states in the model. In fact, we can check that there are 5 stalling states in which the pilot believes in a stall with a degree of less than 1.5%. These are very interesting configurations that capture what may have happened on board of AF447: in these 5 states, the speed sensor is faulty (as a result of the Pitot tubes being frozen) and may report wrong measures, the attitude is UP, the speed is very low, and as a result of this low speed the stall warning remains silent. Notice that, in these specific cases, modifying the attitude to descend results in an increase in speed of the airplane, therefore re-starting the stall warning in the cabin: this is even more confusing for the pilot, as a manoeuvre that reduces the likelihood of stalling in fact generates a stall warning!

The generation of all the discretized states and its encoding as a Mc-COGWED input file require less than a minute, and Mc-COGWED can verify the formula encoding situational awareness for the stall situation in less than 8 seconds.

We argue that the doxastic pattern above can be used to characterise (the lack of) situational awareness in the general

case: the formula

$$\varphi \rightarrow B_{<\delta}^i \varphi$$

is true in states in which  $\varphi$  holds, but agent  $i$  has a degree of belief less than  $\delta$  that this is indeed the case. The parameter  $\delta$  could be configured depending on the specific domain, and can be interpreted as a measure of *situational awareness*.

In the AF447 scenario, it is interesting to see how the situational awareness of a stall could be *increased*. The disengagement of the stall warning at low speed is justified by the necessity of performing low-speed operations close to the ground and to avoid spurious warnings, for instance when taking off or while landing; this, however, results in the pilot not being able to diagnose a stall at very low speed in other conditions. To address this issue, an additional visual indicator of stall warning with low speed readings could be added to the cockpit: this would be similar to ABS warnings on certain car models that remain active under 10 MPH. The additional indicator would reduce the number of possible worlds that the pilot considers possible, thereby *increasing* the minimum value of  $\delta$  for which the formula above is true. This is exactly in line with the recommendations of the BAE to modify the stall management procedures on Airbuses, by re-designing the Primary Flight Display output and by adding additional training requirements in high-altitude stalling conditions.

## 6. RELATED WORK

Formalisms to model degrees of belief have been investigated in the past by a number of authors. Dempster-Shafer belief functions [22] are among the most common approaches to assign a *mass* to beliefs and to combine belief functions. This formalism is a classical example of *subjective* assignment in which *plausibility* can be modelled differently from *probability*. Due to space limitations, we refer to [13] for other approaches to modelling degrees of belief subjectively. In all these formalisms, however, the function associating a weight to a belief needs to be externally provided, for instance by employing historical data or other means; this is a key difference with our approach, where degrees are computed as the ratio between two sets of possible worlds.

The idea of evaluating degrees of belief as the ratio between possible worlds is not new: in the formalism of *random worlds* [2] degrees of belief are computed using *proportion expressions* of the form  $||\varphi(x)||\psi(x)||$ . These expressions denote the proportion of domain elements satisfying  $\varphi$  w.r.t. those satisfying  $\psi$  in the domain of a knowledge base. Conditional expressions are used in [2] to evaluate the weight of beliefs in knowledge bases and are shown to satisfy a set of *desiderata* for default reasoning. While “computationally grounded” in the sense that degrees of belief are not provided externally, computing degrees of belief using random worlds is an undecidable problem in the general case. Moreover, there does not seem to be a tractable solution to add temporal reasoning to this formalism as we do here (as exemplified in the case of the dining cryptographers). Additionally, another key difference with our approach is that we provide a *formal language* to express degrees of belief for a *system* of agents and we are not limited to the single agent case. Along similar lines, the work in [11] introduces *plausibility measures* that are used to justify a set of axioms for default reasoning. More recently, the work in [14] addresses decision making in terms of weighted sets of probabilities

by introducing an axiomatization and by providing *dynamic* decision making procedures.

A language that combines first-order logic and probability in finite domains is introduced in [21] using *Markov Logic Networks* (MLN): similarly to [2], knowledge bases are employed as the underlying semantics, and weights are associated to formulae in the KB. In the case of finite domains weights can be learned using a set of algorithms and the authors show that MLN can tackle real scenarios. The work in [7] presents the logic *PFKD45*, whose syntax is very similar to COGWED. The semantics of this logic relies on externally-provided probability measures over finite bases; the authors present an axiomatization and a decision procedure for this logic but no model checking algorithm. The key differences with our work are the different semantics based on interpreted systems and the inclusion of multiple agents and temporal modalities, in addition to a dedicated model checking tool.

In the multi-agent system community there have been a number of works addressing the verification of doxastic modalities, such as the Jason tool [3] and the AIL+AJPF framework [8]. These two works address BDI architectures and are capable of verifying “standard” (i.e., non-weighted) doxastic operators. The tool MCK [12] has recently been extended to include probabilistic reasoning. In this tool probabilities are assigned to *temporal relations*; the tool is able to verify only the probability of Boolean expressions, possibly nested in an X (next-state) temporal operator. Probabilities over temporal relations are also analysed using the logic PCTL in the well known tool PRISM [17], which has recently been extended to verify probabilistic ATL [5]. A logic to reason about probabilistic knowledge and strategies is also described in [15]: in this work probabilities are associated to temporal relations and to *observations* as well. Our key difference is again in the definition of degrees of belief in terms of *possible worlds*.

More importantly, the PRISM and MCK tool and the approach in [15] all employ *probabilities over transitions*. As mentioned in the introduction, we refer instead to *degrees of belief*. The relationship between these two concepts has been investigated in [2] for a scenario very similar to ours, where degrees are computed as the ratio between two sets of possible worlds. Similarly to this work, in our setting all the possible worlds are equally likely and we do not model probabilities of *transitions*. Essentially, our approach adopts the *principle of indifference* by Bernoulli and Laplace. As described in [2], a uniform distribution for possible worlds is the one that maximizes entropy. In turn, this corresponds to the least amount of *information* about the probability distribution of epistemically equivalent worlds. In other words, we start from an *unknown* objective assignment of probabilities to transitions and we build a *subjective* assignment of degrees of belief to agents according to this unknown objective assignment; agents’ degrees of belief can then be interpreted using a computationally grounded evaluation.

## 7. CONCLUSION

In this paper we have introduced COGWED, an extension of CTLK to reason about degrees of belief in a system of agents. We have introduced a computationally grounded semantics based on Interpreted Systems, we have presented a model checking algorithm for COGWED and we have investigated its complexity, showing that model checking CO-

WED has the same complexity of model checking CTLK. To validate our claims, we have implemented and made publicly available Mc-COGWED, a Java-based explicit state model checker, and we have assessed the performance of our algorithm against the standard benchmark of the dining cryptographers. The results obtained are very encouraging: our prototype was able to verify up to 15 cryptographers, a figure comparable to state-of-the-art model checkers for multi-agent systems.

To prove the applicability of COGWED to real scenarios we have collaborated with domains experts at NASA Ames to assess the *situational awareness* of aircraft pilots flying in off-nominal conditions, obtaining results that are in line with BAE recommendations. Finally, we have presented a detailed review of related work, highlighting our contributions and discussing the relationship between degrees of belief as modelled in COGWED and probability measures over temporal transitions.

As mentioned in the previous section, our approach considers all the possible worlds equally likely: this is the result of *ignoring* the probability distribution of temporal transitions. We are currently working at incorporating this information into the doxastic characterisation of agents. In particular: what can be said when the pilot *knows* that a certain sensor has a higher probability of failure than another sensor? What could be said about the resulting degrees of belief?

## 8. REFERENCES

- [1] A. Agogino and G. Brat. Statistical analysis of flight procedures. Technical report, NASA Ames Research Center, Moffett Field, Mountain View (CA), 2011.
- [2] Fahiem Bacchus, Adam J. Grove, Joseph Y. Halpern, and Daphne Koller. From statistical knowledge bases to degrees of belief. *Artificial Intelligence*, 87:75–143, 1996.
- [3] Rafael H. Bordini, Jomi Fred Hübner, and Michael Wooldridge. *Programming Multi-Agent Systems in AgentSpeak using Jason (Wiley Series in Agent Technology)*. John Wiley & Sons, 2007.
- [4] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.
- [5] T. Chen, V. Forejt, M. Kwiatkowska, D. Parker, and A. Simaitis. PRISM-games: A model checker for stochastic multi-player games. In N. Piterman and S. Smolka, editors, *Proc. 19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'13)*, volume 7795 of *LNCS*, pages 185–191. Springer, 2013.
- [6] Edmund M Clarke, Orna Grumberg, and Doron A Peled. *Model checking*. MIT press, 1999.
- [7] Nivea de Carvalho Ferreira, Michael Fisher, and Wiebe Van Der Hoek. Specifying and reasoning about uncertain agents. *International Journal of Approximate Reasoning*, 49(1):35–51, 2008.
- [8] Louise A Dennis, Michael Fisher, Matthew P Webster, and Rafael H Bordini. Model checking agent programming languages. *Automated Software Engineering*, 19(1):5–63, 2012.
- [9] Ronald Fagin, Joseph Y Halpern, Yoram Moses, and Moshe Y Vardi. *Reasoning about knowledge*. MIT press Cambridge, 1995.
- [10] Han Tin French, Elizabeth Clarke, Diane Pomeroy, Melanie Seymour, and C Richard Clark. Psycho-physiological measures of situation awareness. *Decision Making in Complex Environments*, page 291, 2007.
- [11] Nir Friedman and Joseph Y Halpern. Plausibility measures and default reasoning. *Journal of the ACM*, 48(4):648–685, 2001.
- [12] Peter Gammie and Ron Van Der Meyden. Mck: Model checking the logic of knowledge. In *Computer Aided Verification*, pages 479–483. Springer, 2004.
- [13] Joseph Y Halpern. *Reasoning about uncertainty*. The MIT Press, 2003.
- [14] Joseph Y. Halpern and Samantha Leung. Weighted sets of probabilities and minimaxweighted expected regret: New approaches for representing uncertainty and making decisions. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, August 14–18, 2012*, pages 336–345, 2012.
- [15] Xiaowei Huang and Cheng Luo. A logic of probabilistic knowledge and strategy. In *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '13, Saint Paul, MN, USA, May 6–10, 2013*, pages 845–852, 2013.
- [16] Josie Hunter, Franco Raimondi, Neha Rungta, and Richard Stocker. A synergistic and extensible framework for multi-agent system verification. In *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '13, Saint Paul, MN, USA*, pages 869–876, 2013.
- [17] Marta Kwiatkowska, Gethin Norman, and David Parker. Prism 4.0: Verification of probabilistic real-time systems. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Computer Aided Verification*, volume 6806 of *Lecture Notes in Computer Science*, pages 585–591. Springer Berlin Heidelberg, 2011.
- [18] Alessio Lomuscio, Hongyang Qu, and Franco Raimondi. Mcmas: A model checker for the verification of multi-agent systems. In Ahmed Bouajjani and Oded Maler, editors, *Computer Aided Verification*, volume 5643 of *Lecture Notes in Computer Science*, pages 682–688. Springer Berlin Heidelberg, 2009.
- [19] Alessio Lomuscio and Franco Raimondi. The complexity of model checking concurrent programs against ctlk specifications. In *Declarative Agent Languages and Technologies IV*, pages 29–42. Springer, 2006.
- [20] Terence Parr. *The Definitive ANTLR Reference: Building Domain-Specific Languages*. Pragmatic Bookshelf, 2007.
- [21] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62(1-2):107–136, 2006.
- [22] Glenn Shafer. *A mathematical theory of evidence*, volume 1. Princeton university press Princeton, 1976.
- [23] Michael Wooldridge. Computationally grounded theories of agency. In *Proceedings of ICMAS, International Conference of Multi-Agent Systems*, pages 13–20. IEEE Press, 2000.